The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 6, pp.113-119, Dec. 31, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.6.113
JIIBC 2022-6-17

딥러닝을 이용한 소프트웨어 결함 심각도 예측

Prediction of Software Fault Severity using Deep Learning Methods

홍의석*

Euyseok Hong*

요 약 소프트웨어 결함 예측 작업 시 단순히 결함 유무만을 예측하는 이진 분류 모델에 비해 결함의 심각도 범주를 예측하는 다중 분류 모델은 훨씬 유용하게 사용될 수 있다. 소수의 심각도 기반 결함 예측 모델들이 제안되었지만 딥러닝 기법을 사용한 분류기는 없었다. 본 논문은 3개, 5개의 은닉층을 갖고 은닉층 노드수가 고정된 구조와 변화하는 구조의 MLP 모델들을 제작하였다. 모델 평가 실험 결과 기존 기계학습 모델들 중 가장 좋은 성능을 보인 MLPs보다 MLP 기반 딥러닝 모델들은 Accuracy와 AUC 모두 유의미하게 더 우수한 성능을 보였다. 특히 노드수 고정 구조에서는 은닉층수 3, 배치사이즈 32, 노드수 64인 모델 구조가 가장 좋은 성능을 보였다.

Abstract In software fault prediction, a multi classification model that predicts the fault severity category of a module can be much more useful than a binary classification model that simply predicts the presence or absence of faults. A small number of severity-based fault prediction models have been proposed, but no classifier using deep learning techniques has been proposed. In this paper, we construct MLP models with 3 or 5 hidden layers, and they have a structure with a fixed or variable number of hidden layer nodes. As a result of the model evaluation experiment, MLP-based deep learning models shows significantly better performance in both Accuracy and AUC than MLPs, which showed the best performance among models that did not use deep learning. In particular, the model structure with 3 hidden layers, 32 batch size, and 64 nodes shows the best performance.

Key Words : Software fault severity, Quality prediction, Deep learning

Ⅰ. 서 론

수십년간 매우 많은 연구가 이루어진 결함 예측 모델들은 대부분 결함경향성 유무만을 예측하는 이진 분류 모델들이었다. 하지만 이진 분류 모델의 문제점은 결함이 가진 심각도 또는 우선순위도 등의 특성을 고려하지 않고 모든 결함을 같은 정도의 문제들이라고 규정한 것이다. 결함 심각도란 시스템 및 사용자들에게 결함이 미치는 충격의 정도이다. 심각도 수준이 매우 높은 결함은 시스템을 중단시킬 정도이며, 매우 낮은 결함은 시스템에 거의 영향을 미치지 않는다. 따라서 심각도 수준에 따라 결함을 예측하는 다중 분류 모델은 고심각 결함 부분

*정회원, 성신여자대학교 컴퓨터공학과
접수일자 2022년 10월 2일, 수정완료 2022년 11월 2일
게재확정일자 2022년 12월 9일

Received: 2 October, 2022 / Revised: 2 November, 2022 / Accepted: 9 December, 2022
*Corresponding Author: hes@sungshin.ac.kr
Dept. of Computer Engineering, Sungshin Women's University, Korea

에 적절한 자원을 할당함으로써 결함 유무 예측 모델보다 훨씬 유용하게 사용될 수 있다[1].

예측 모델 구축에 가장 많이 사용되는 기법인 기계학습 분야에서 가장 각광 받는 딥러닝 기법은 소프트웨어 결함 예측 분류기 제작에는 거의 사용되지 않았다. 그 이유는 학습 데이터의 작은 사이즈 및 특성이 딥러닝 학습에 적합하지 않았기 때문이다. 오히려 딥러닝은 소스 코드나 설계 결과들을 이용해 여러 시맨틱한 정보를 갖춘 학습 데이터를 제작하는데 더 많이 사용되었다[2].

본 논문의 목적은 딥러닝 기법으로 심각도 기반 결함 예측 모델을 제작해 딥러닝을 사용하지 않는 모델들과 성능을 비교해 보는 것이다. 또한 다중 분류 예측 모델에서 딥러닝이 기존 기계학습 기법들보다 의미있는 결과를 내는지도 알아볼 것이다. 사용하는 딥러닝 기법은 특정 데이터 학습에 주로 사용되는 CNN이나 RNN이 아닌 은닉층이 2개 이상인 MLP 모델이며 본 논문에서는 이를 DNN(Deep Neural Networks), 은닉층이 하나인 기존 신경망 모델을 MLPs(MLP with single hidden layer)라 지칭한다.

II. 결함 심각도 예측

결함 심각도는 심각한 수준을 나타내는 레벨값으로 표현된다. IEEE Std. 1044-2009에서는 낮은 레벨에서 높은 레벨값으로 Inconsequential에서 Blocking까지 5단계로, Bugzilla에서는 Enhancement에서 Blocking까지 7단계로 나타내었다. 결함 예측 모델 연구에 많이 사용된 NASA의 MDP 공개 데이터 집합들 중 몇 개의 프로젝트는 5단계의 심각도 레벨값을 가지고 있으며 심각도 1이 가장 심각한 결함을 나타낸다.

심각도 예측 모델들에 관한 연구들은 모두 딥러닝을 사용하지 않은 감독형 모델들이다. [1], [3]은 NASA 데이터 집합의 KC1 프로젝트를 사용하였다. [3]은 심각도 1을 고심각, 2~5를 저심각 결함으로 결함을 두 범주로 나누었으며 [1]은 심각도 2를 중심각으로 둠으로써 세 개의 범주로 나누었다. 두 연구 모두 출력이 여러개로 나뉘는 다중 분류 모델을 구현하지 않고 예측하고자하는 부류와 나머지 부류로 나누어 예측하는 이진 분류 모델 형태를 실험에 사용하였다. 그 결과 고심각 보다는 저심각이나 중심각의 예측이 좋은 성능을 보였다. 모델 구축 알고리즘들은 NB(Naïve Bayes), Random Forest, 판단트리, MLPs 등을 사용하였다. [4]는 Bugzilla를 사용한 Eclipse 프로젝트의 세 개의 공개 릴리즈 데이터 집합을 사용하여 이전 릴리즈 데이터로 학습한 모델이 다음 릴리즈 데이터의 심각도를 예측할 수 있는지를 연구하였다. 7개의 심각도 레벨에서 가장 낮은 두 개를 제외하고 세 개의 범주로 결함을 나누었으며, 릴리즈가 반복될수록 예측 성능은 점점 감소함을 보였다. [5]는 NASA 데이터 집합 중 JM1, PC4를 사용하여 MLPs, NB, J48(판단트리)의 성능을 비교하였다. 결함은 두 범주로 분류하고 비결함까지 3개의 출력을 갖는 삼중 분류 모델을 만들었으며, 실험 결과 MLPs가 가장 좋은 결과를 보였다. [6]은 [5]의 실험 조건에서 추가로 AUC, FNR을 구하였으며 매우 안좋은 FNR 결과의 이유로 출력 클래스의 불균형 문제를 제시하였다. 그에 대한 해결책으로 샘플링 기법들을 사용하여 불균형 데이터 처리 모델들을 제작하였으며 SMOTE 모델이 가장 좋은 성능을 보임을 보였다.

본 논문은 [5]와 같은 데이터를 사용하여 제작한 DNN의 성능과 [5]의 결과 모델의 성능을 비교 평가할 것이다.

III. 모델 셀계

1. 출력 클래스 결정

NASA 공개 데이터 집합은 초기 버전과 PROMISE 버전이 존재한다. 이들 중 심각도 정보가 있는 초기 버전의 프로젝트는 여러개의 파일로 구성되는데, 모듈의 입력 메트릭 집합과 심각도 정보는 다른 파일에 분리되어 있으므로 이들을 합성하여 사용하였다[5].

5단계의 심각도 값을 모두 출력 클래스로 지정하면 비결함 출력까지 총 6개의 모델 출력이 존재한다. 타 연구들과 같이 범주를 줄여 결함 클래스를 두 개로 나누었다. 따라서 출력 클래스는 HSF(High Severity Fault-prone), LSF(Low Severity Fault-prone), NF(Not Fault-prone)가 된다. 또한 고심각에 대한 두가지 해석을 하여 레벨 1인 것만 고심각으로 보는 SEVERITY1과 레벨 1, 2를 고심각으로 보는 SEVERITY2 데이터를 준비하였다.

표 1. 예측 모델의 심각도 출력
Table 1. Severity output of the prediction model

데이터종류	HSF	LSF	NF
SEVERITY1	심각도 1	심각도 2,3,4,5	결함 없음
SEVERITY2	심각도 1,2	심각도 3,4,5	결함 없음

2. 모델 구조

모델 구조 중 가장 중요한 것은 은닉층의 수와 각 층의 노드수를 결정하는 것으로 최적값의 결정은 여러 실험과 경험에 의해 이루어진다. 본 논문은 JM1을 사용한 이진 분류 DNN 모델 연구인 [7]의 모델 구조와 하이퍼 파라미터 셋팅을 상당 부분 사용한다.

모델 구조는 크게 은닉층 노드수를 고정하는 구조와 출력층으로 갈수록 노드수를 줄여가는 구조로 나눌 수 있다. 데이터 규모가 크지 않으므로 은닉층수는 3, 5로 하였고 노드수 고정 구조에서는 노드수를 32, 48, 64로 하는 모델을 제작하였다. 노드수 변화 구조에서는 은닉층수 3인 경우는 64-48-32, 5인 경우는 입력 노드수에 따라 64-64-48-32-16, 64-64-48-48-32 등으로 하였다. PC4는 입력 노드가 36개로 JM1의 21개보다 크다. 그림 1, 그림 2는 입력 노드가 p개인 경우 두 구조를 나타낸다.

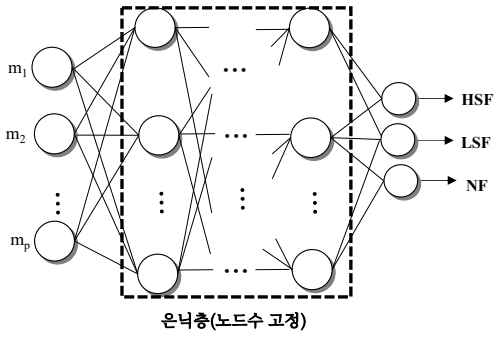

그림 1. 노드수 고정 모델 구조
Fig. 1. Model structure with fixed number of nodes

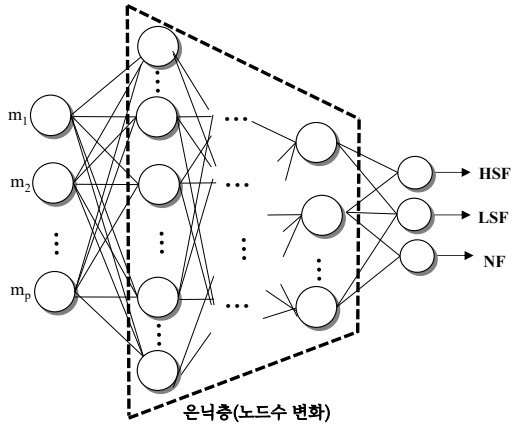

그림 2. 노드수 변화 모델 구조
Fig. 2. Model structure with changing number of nodes

하이퍼 파리미터도 모델 구조와 같이 경험 또는 휴리스틱한 방법을 통해 결정된다. 가중치 변화 방법인 미니 배치 기법에서 미니 배치에 넘겨주는 데이터 개수인 배치 사이즈는 5, 10, 32로 설정하였다. 전체 데이터 학습 및 테스트의 반복 횟수인 에포크 수는 모델의 과적합을 막기 위해 600을 상한을 정하고 Keras의 EarlyStopping 함수를 사용하였다. 또 다른 과적합을 막기위한 드롭아웃 기법도 사용하였으며 드롭아웃 비율은 0.5로 하였다. 손실함수는 다중 분류 모델에서 일반적으로 사용되는 categorical crossentropy 함수를 사용하였고, 최적화 함수는 adam, 활성화 함수는 은닉층에서는 ReLU 함수를, 출력층에서는 sigmoid 함수를 사용하였다.

Ⅳ. 실험 및 결과

1. 평가 척도

모델의 성능 평가를 위한 척도로는 ACC(Accuracy)와 AUC를 사용한다. ACC는 다중 분류 모델의 경우에도 구할 수 있지만 AUC는 이진 분류 모델의 척도이다. 따라서 AUC는 가장 중요한 출력 클래스인 HSF와 나머지(LSF와 NF)를 이진 분류 모델로 가정하여 구한다. AUC는 ACC가 결함 데이터의 불균형성을 평가 못한다는 단점을 보완하는 척도로 1에 가까울수록 높은 성능을 나타내며 0.7 이상의 모델은 사용가능한 모델로 분류된다[8]. 입력 데이터의 속성선정을 통해 차원축소가 필요한 경우는 cfs(correlation based feature selection)를 사용하였다.

2. 기존 모델 성능 평가 실험

표 2와 표 3은 딥러닝을 사용하지 않은 기존 연구의 실험 결과들을 나타낸 것이다. 표 2는 JM1, PC4에 대해 ACC 측정 실험만을 한 결과이고 as는 차원축소한 데이터를 의미한다[5]. MLPs 결과만을 볼 때 차원축소 여부의 영향은 거의 없었으며 PC4에 대한 결과가 JM1보다 훨씬 좋았다. 표 3은 JM1에 대해 ACC와 함께 AUC를 측정한 결과이다[6]. ACC는 표 2와 차이가 거의 없었으며 AUC는 MLPs와 NB가 J48보다 나은 결과를 보였다.

결과적으로 기존 실험에서는 ACC와 AUC 모두 MLPs가 가장 좋은 성능을 보였으므로 본 연구에서는 딥러닝을 사용한 DNN과 MLPs의 성능을 비교할 것이다.

표 2. 딥러닝을 사용하지 않은 모델 실험결과 1
Table 2. Experimental results 1 without deep learning

데이터종류	데이터집합	MLPs	NB	J48
SEVERITY1	JM1	0.8075	0.7870	0.7851
	JM1_as	0.8073	0.7864	0.7998
	PC4	0.8797	0.6480	0.8685
	PC4_as	0.8805	0.8621	0.8807
SEVERITY2	JM1	0.8075	0.7865	0.7829
	JM1_as	0.8077	0.7844	0.7939
	PC4	0.8843	0.4995	0.8640
	PC4_as	0.8895	0.8855	0.8861

표 3. 딥러닝을 사용하지 않은 모델 실험결과 2
Table 3. Experimental results 2 without deep learning

데이터종류	성능 척도	속성 선정	MLPs	NB	J48
SEVERITY1	ACC	all	0.81	0.79	0.78
		cfs	0.81	0.79	0.80
	AUC	all	0.76	0.76	0.53
		cfs	0.76	0.75	0.60
SEVERITY2	ACC	all	0.81	0.79	0.78
		cfs	0.81	0.78	0.79
	AUC	all	0.68	0.68	0.55
		cfs	0.69	0.68	0.56

3. 딥러닝 모델 성능 평가 실험

입력 데이터의 차원축소 여부가 DNN 성능에 미치는 영향을 알아보기 위하여 SEVERITY1-JM1에 대해 속성 선정을 통해 차원축소를 한 경우와 모든 입력 데이터를 사용한 경우의 비교 실험을 진행하였다. 실험 모델은 여러 모델 구조들 중 가장 좋은 성능을 보이는 은닉층수 3, 배치 사이즈 32, 노드수는 64인 노드수 고정 구조를 사용하였다. 표 4는 해당 실험 결과로 차원축소 여부는 모델 성능에 유의미한 차이를 만들지 않았다. 오히려 차원축소를 안한 경우가 미세하게나마 차원축소한 경우보다 ACC, AUC 모두 좋은 결과를 보였다. 따라서 성능 평가 실험은 차원축소 작업을 안한 전체 입력 데이터를 사용하여 수행하였다.

표 4. 속성선정 작업의 효과 실험결과
Table 4. Effect Experimental results of feature selection

은닉 층수	배치 사이즈	노드수	속성 선정	Loss	ACC	AUC
3	32	64	all	0.5382	0.8132	0.7774
			cfs	0.5172	0.8104	0.7690

JM1, PC4 두 실험 데이터 집합이 SEVERITY1, SEVERITY2에 대해 다른 출력값을 가지므로 실험 결과는 총 4개가 된다. 표 5와 표 6은 SEVERITY1에 대한 JM1, PC4의 결과이고 표 7과 표 8은 SEVERITY2에 대한 JM1, PC4의 결과이다. 각 표는 은닉층의 노드수 고정 구조와 변화 구조의 결과들을 모두 나타내었다. 결과에서 진하게 표시한 부분은 해당 표에서 가장 좋은 결과를 나타내는 부분이다. 표 5는 SEVERITY1-JM1의 결과로, 노드수를 고정한 모델 구조에서는 은닉층수, 배치 사이즈, 노드수를 3, 32, 64로 한 경우의 결과가 가장 좋았다. 노드수를 변화시키는 경우는 은닉층이 3개인 경우와 5개인 경우 가장 좋은 결과를 나타낸 64-48-32와 64-64-48-32-16의 결과를 나타내었다. 노드수 변화 모델의 가장 좋은 결과인 은닉층 3개인 경우가 고정인 경우보다 ACC는 미세하게 높았지만 AUC는 미세하게 낮았다. 딥러닝을 사용하지 않은 MLPs의 ACC, AUC는 대략 0.8075와 0.76으로 DNN이 MLPs보다 각각 0.7~0.9%, 2.3% 정도 성능이 상승하였다.

표 5. SEVERITY1-JM1 실험결과
Table 5. Experimental results of SEVERITY1-JM1

은닉 층수	배치 사이즈	노드수/ 노드수변화	Loss	ACC	AUC
3	5	32	0.5483	0.8086	0.7605
		48	0.6436	0.8074	0.7579
		64	0.5594	0.8069	0.7645
	10	32	0.5388	0.8087	0.7643
		48	0.5300	0.8084	0.7664
		64	0.5474	0.8093	0.7493
	32	32	0.5182	0.8099	0.7680
		48	0.5158	0.8116	0.7720
		64	0.5382	**0.8132**	**0.7774**
5	5	32	0.6633	0.8070	0.7610
		48	0.6105	0.8069	0.7644
		64	0.7232	0.8069	0.7586
	10	32	0.6199	0.8071	0.7641
		48	0.6145	0.8068	0.7682
		64	0.5972	0.8070	0.7661
	32	32	0.5907	0.8067	0.7573
		48	0.5578	0.8073	0.7627
		64	0.5938	0.8073	0.7341
3	32	64-48-32	0.5197	**0.8148**	**0.7745**
5	32	64-64-48 -32-16	0.5281	0.8098	0.7615

표 6은 SEVERITY1-PC4의 실험 결과로 가장 좋은 결과를 보인 모델 구조는 노드수 고정의 경우와 변화의 경우 모두 JM1의 경우와 같았다. 심지어 노드수 변화 모델이 노드수 고정 모델의 최고 결과보다 ACC값은 좀 더

크고, AUC값은 좀 더 작은 특징도 같았다. PC4는 입력 메트릭이 36개로 JM1의 21개보다 많이 크다. 따라서 노드수 변화 모델에서 64-64-48-48-32로 한 경우가JM1의 경우인 64-64-48-32-16 보다 좋은 결과를 보였다. 표 3의 기존 연구는 JM1에만 행해졌기 때문에 PC4에 대한 MLPs의 AUC 결과값은 없다. 따라서 표 2의 ACC 결과인 0.8797과 비교하면 DNN은 MLPs에 비해 노드수 고정 모델이 4.24%, 노드수 변화 모델이 5.01% 더 좋은 성능 결과를 보였다.

표 6. SEVERITY1-PC4 실험결과
Table 6. Experimental results of SEVERITY1-PC4

은닉층수	배치사이즈	노드수/노드수변화	Loss	ACC	AUC
3	5	32	0.8792	0.9033	0.9402
		48	1.3820	0.9019	0.9411
		64	0.9235	0.9136	0.9431
	10	32	0.6548	0.8998	0.9328
		48	0.6781	0.9101	0.9525
		64	0.4582	0.9122	0.9496
	32	32	0.5672	0.9033	0.9437
		48	0.3529	0.9142	0.9542
		64	0.4643	**0.9170**	**0.9595**
5	5	32	11.008	0.8779	0.8931
		48	7.2515	0.8854	0.9257
		64	3.9351	0.8951	0.9407
	10	32	1.2409	0.8950	0.9373
		48	2.3529	0.8834	0.9370
		64	2.4550	0.9019	0.9479
	32	32	0.6435	0.9047	0.9479
		48	0.4435	0.8971	0.9497
		64	0.5430	0.9033	0.9539
3	32	64-48-32	0.4772	**0.9238**	**0.9573**
5	32	64-64-48-48-32	0.4719	0.9012	0.9514

표 7은 SEVERITY2-JM1에 대한 DNN의 실험 결과로 SEVERITY1-JM1보다 불규칙적인 결과를 보인다. 가장 좋은 결과는 은닉층 3개의 노드수 변화 모델로 노드수 고정 모델들보다 더 좋은 성능을 보였다. 노드수 고정 모델들에서는 유의미하게 가장 좋은 결과를 내는 모델은 없었다. SEVERITY1에서 가장 좋은 결과를 낸 구조인 은닉층수 3, 배치사이즈 32, 노드수 64 모델은 나름 좋은 결과를 보였지만 타 구조와 성능 차이가 매우 작았고 유사한 성능을 보인 구조가 2~3개 더 있었다. MLPs의 ACC, AUC인 0.8075, 0.68과 비교하면 노드수 고정 모델은 미세한 성능 증가가 있었고, 노드수 변화 모델은 0.33%, 2.84%의 성능 증가가 있었다.

표 7. SEVERITY2-JM1 실험결과
Table 7. Experimental results of SEVERITY2-JM1

은닉층수	배치사이즈	노드수/노드수변화	Loss	ACC	AUC
3	5	32	0.6205	0.8074	0.6885
		48	0.6004	0.8076	0.6877
		64	0.5517	0.8085	0.6928
	10	32	0.5477	0.8076	0.6914
		48	0.5463	0.8087	0.6943
		64	0.5502	0.8084	0.6896
	32	32	0.5373	0.8083	0.6926
		48	0.6002	0.8076	0.6877
		64	0.5517	0.8085	0.6928
5	5	32	0.6184	0.8071	0.6838
		48	0.6512	0.8068	0.7629
		64	0.5545	0.8067	0.6890
	10	32	0.5694	0.8070	0.6891
		48	0.5419	0.8070	0.6886
		64	0.5482	0.8071	0.6904
	32	32	0.5641	0.8074	0.6911
		48	0.5560	0.8083	0.7683
		64	0.6149	0.8074	0.6770
3	32	64-48-32	0.5455	**0.8102**	**0.6993**
5	32	64-64-48-32-16	0.5473	0.8080	0.6917

표 8. SEVERITY2-PC4 실험결과
Table 8. Experimental results of SEVERITY2-PC4

은닉층수	배치사이즈	노드수/노드수변화	Loss	ACC	AUC
3	5	32	0.5488	0.9094	0.9614
		48	0.6037	0.9087	0.9586
		64	0.5682	0.9238	0.9724
	10	32	0.3930	0.9143	0.9655
		48	0.3446	0.9238	0.9734
		64	0.4160	0.9273	0.9727
	32	32	0.5076	0.9060	0.9603
		48	0.3227	0.9245	0.9707
		64	0.3117	**0.9300**	**0.9749**
5	5	32	1.8639	0.9080	0.9493
		48	5.1324	0.8991	0.9427
		64	1.8997	0.9053	0.9264
	10	32	2.2737	0.8847	0.9401
		48	0.6641	0.9135	0.9619
		64	1.2111	0.9087	0.9515
	32	32	0.4545	0.9005	0.9515
		48	0.6171	0.9046	0.9509
		64	0.4369	0.9040	0.9561
3	32	64-48-32	0.4601	0.9163	0.9716
5	32	64-64-48-48-32	0.9355	0.9170	0.9498

PC4의 경우는 SEVERITY1의 경우와 같이 은닉층수 3, 배치사이즈 32, 노드수 64인 경우에 ACC과 AUC 결

과가 가장 좋았다. 단, 다른 세 경우와는 다르게 노드수 변화 모델보다 노드수 고정 모델의 결과가 성능이 더 좋았다. 노드수 변화 모델에서는 은닉층수 3과 5인 모델 중 더 나은 모델을 정하기는 어려웠다. MLPs의 ACC인 0.8843과 비교하면 노드수 고정 모델은 5.17% 성능 향상을 보였고 노드수 변화 모델도 그보다는 작지만 성능 향상을 보였다.

4. 이상치 데이터 제거 실험

이상치 데이터를 제거하여 모델의 성능을 높이는 실험을 수행하였다. JM1에 [7]에서 사용한 IQR(Inter Quantile Range) 기법을 적용하여 데이터 분포에서 많이 벗어난 이상치들을 찾아내었다. 779개의 이상치 데이터를 제거하였으며 결과적으로 10,099개의 정제된 데이터를 사용하여 성능 실험을 하였다.

표 9. 이상치 데이터 제거 실험결과
Table 9. Experimental results when outlier data is removed

은닉층수	배치사이즈	노드수/노드수변화	Loss	ACC	AUC
3	5	32	0.4794	0.8379	0.7933
		48	0.5016	0.8379	0.7859
		64	0.5009	0.8359	0.7928
	10	32	0.4737	0.8392	0.7922
		48	0.4851	0.8385	0.7954
		64	0.4746	0.8414	0.7975
	32	32	0.4689	0.8398	0.7948
		48	0.4657	0.8407	0.8006
		64	0.4661	0.8401	0.7968
5	5	32	0.4951	0.8379	0.7557
		48	0.4955	0.8362	0.7686
		64	0.5002	0.8356	0.7336
	10	32	0.4805	0.8385	0.7972
		48	0.4844	0.8399	0.7857
		64	0.4821	0.8373	0.7999
	32	32	0.5020	0.8384	0.7980
		48	0.4736	0.8383	0.7992
		64	0.4733	0.8396	0.7980
3	32	64-48-32	0.4692	0.8388	0.7975
5	32	64-64-48-32-16	0.4744	0.8393	0.7956

SEVERITY1-JM1으로 실험한 결과, 모든 모델 구조에서 이상치 데이터 제거 전보다 좋은 성능을 보였다. ACC, AUC 모두 거의 모든 모델 구조에서 0.03 정도 성능이 상승하였고, Loss는 하락하였다. ACC와 AUC의 최고값은 딥러닝을 사용하지 않은 MLPs보다 각각 4.2%, 5.34%의 높은 성능 향상을 보였다.

V. 결 론

결함 유무만을 판단하는 일반적인 이진 분류 구조의 결함 예측 모델보다 결함 심각도에 기반한 다중 분류 모델은 매우 심각한 결함 부분을 예측할 수 있다는 점에서 매우 유용하다. 본 연구는 심각도 기반 결함 예측 모델들을 딥러닝 기법을 이용하여 구현하였으며, 구현 모델들의 성능을 하나의 은닉층을 갖는 신경망 모델인 MLPs와 비교하였다.

제안 모델들은 은닉층의 노드수 고정 모델과 노드수 변화 모델로 나뉠 수 있다. 실험은 NASA 공개 데이터 집합들 중 JM1, PC4를 이용하였으며 고심각 결함의 해석에 따른 두개의 데이터 종류에 대해 실시하였다. 실험 결과는 4가지 다른 데이터 경우 모두 딥러닝 모델이 딥러닝을 사용하지 않은 MLPs보다 좋은 성능을 보였다. 특히 노드수 고정 모델에서는 은닉층수 3, 배치사이즈 32, 노드수 64인 모델 구조가 Accuracy와 AUC 모두 높은 성능을 보였다. 노드수 변화 모델에서도 은닉층이 3개인 구조가 5개인 구조보다 나은 성능을 보였으며 Accuarcy는 노드수 고정 모델보다 약간 나은 결과를 보였지만 AUC는 그렇지 않았다. IQR 기법을 사용하여 이상치 데이터를 제거한 실험에서는 모델 성능이 탁월하게 좋아졌다.

전체적으로 딥러닝 모델은 JM1에서 딥러닝을 사용하지 않은 모델보다 Accuarcy는 최고 4.2%, AUC는 5.34%의 성능향상이 있었고, 기존 모델의 AUC 결과가 없는 PC4에서는 Accuracy가 최고 5.17% 상승하였다.

References

[1] Y. Singh, A. Kaur and R. Malhotra, “Empirical validation of object oriented metrics for predicting fault proneness models,” Soft. Quality Journal, Vol. 18, pp. 3-35, March 2010.
DOI: https://doi.org/10.1007/s11219-009-9079-6

[2] E. N. Akimova et al., "A survey on software defect prediction using deep learning", Mathematics, vol. 9, no. 11, 2021.
DOI: https://10.3390/math9111180

[3] Y. Zhou and H. Leung, “Empirical analysis of object-oriented design metrics for predicting high and low severity faults,” IEEE Trans. Software Eng., Vol.32, No.10, pp.771-789, Oct. 2006.
DOI: https://doi.org/10.1109/tse.2006.102

[4] R. Shatnawi and W. Li, “The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process,” Journal of

Systems and Software, Vol. 81, No. 11, pp. 1868-1882, 2008.
DOI: https://doi.org/10.1016/j.jss.2007.12.794

[5] E. Hong, "Software Quality Prediction based on Defect Severity," Journal of the Korea Society of Computer and Information, Vol. 20, No. 5, pp. 73-81, 2015.
DOI: https://doi.org/10.9708/jksci.2015.20.5.073

[6] E. Hong and M. Park, "Severity-based Software Quality Prediction using Class Imbalanced Data," Journal of the Korea Society of Computer and Information, Vol. 21, No. 4, pp. 73-80, 2016.
DOI: https://doi.org/10.9708/jksci.2016.21.4.073

[7] E. Hong, "Prediction Model of Software Fault using Deep Learning Methods," Journal of the Institute of Internet, Broadcasting and Communication, vol. 21, no. 4, pp. 111-117, 2022.
DOI: https://doi.org/10.7236/JIIBC.2022.22.4.111

[8] T. Fawcett, "An introduction to ROC analysis," Pattern recognition letters, vol. 27, no. 8, pp. 861-874, June 2006.
DOI: https://doi.org/10.1016/j.patrec.2005.10.010

[9] Y. Jo, C. Yoo, and J. Lee, "Clone Type Classification based on Tree-Based Convolution Neural Network," Journal of KIIT, Vol. 18, No. 3, pp. 79-88, 2020.
DOI: http://dx.doi.org/10.14801/jkiit.2020.18.3.79

저 자 소 개

홍 의 석(정회원)

- 1992년 : 서울대학교 계산통계학과 전산과학전공 학사
- 1994년 : 서울대학교 계산통계학과 전산과학전공 석사
- 1999년 : 서울대학교 전산과학과 박사
- 현재 성신여자대학교 컴퓨터공학과 교수
- 관심분야 : 소프트웨어 품질 예측, AI 엔지니어링, MSR 등